

It will be noted that the dam divides the bed into a first and second segment, which are in reverse order, as will be seen in FIG. 14 and 15. This babbling movement may be visualized (FIG. 15) as a data window bounded by twin dams D travelling along the stream. Certain elements accidentally fill the bed-end positions CF, CH, NF, NH as the window moves.

The principle is thus one of incremental movement of the stream through the bed in either direction, with the contents of the bed unceasingly presenting valid and identifiable contents to the display system DI P. The pattern of this movement will best be understood by consideration of FIGS. 14 and 15, where a "window" of stream St is represented in two discontinuous sections 1st and 2^d of Bed B. Filling position F and incrementing register representing divider D has the effect of moving the window one stream position. This is babble as in FIG. 14 BWD and FWD.

It will be noted in FIG. 15 that topologically positions CF and CH do not change until babble reaches them.

Babble, as in FIG. 14, is thus the bed operation of replacing successive elements or groups of elements from the divider in either direction, moving (incrementing) the divider for each such element. The corresponding and simultaneous stream operation, as in FIG. 15, we call roving. Roving is the motion of a current stream/window along a stream/cluster) by continuous increments. Babble and roving are the same operation considered from two points of view; babble is the mechanism by which roving is made possible without clearing and rearranging display lists in a core memory.

The roving operation may also be considered from the point of view of the enfilade, as in FIG. 7, where the "core window" consists of the contents of the bed. Thus to the enfilade, babble and roving are successive sideways motions to data left and right.

This roving feature is obviously useful, especially for data streams of information where sliding sections may be of interest to a viewer. Text, for instance, where a consecutive 1000 characters present a readable subset of a larger written work. The roving feature allows this subset to be moved along the text stream while maintaining continuous visibility to the viewer of the changing display. Picture-drawing data may also be employed in such sliding data windows, to similarly useful effect; so may statistical data where "moving averages" are of use.

DISCONTIGUOUS BEDS AND BUFFERS

A generalized bed structure is desirable when this invention is to be used for a variety of purposes, rather than just for a single purpose such as the Parallel Textface. When the system is to be used for the general communication of text-and-picture complexes, a flexible means of rearranging and re-sizing Beds is required. This we accomplish by means of the Discontiguous Bed or Scatterbed.

A scatterbed is a distributed bed whose parts occupy unconnected sections of core, but which may be babbled as if it were in one piece.

Since "storage bank zero" in the crum structure refers to core memory DCM, the bed's location is then specified as a series of crums. The bed "begins" with the section of core memory specified by the first crum, continues into the section of core memory specified by the second crum, and goes on in this fashion for as many crums as are needed. The divider is stored in rapid-access memory as before.

BUF

9

62

In this case the Babbling algorithms must change in one respect: rather than check for bed overflow in each succeeding element being babbled, they check for overflow of the current section.

5 This method has the obvious advantage of permitting Beds to be readily re-sized as needed. In addition, it retains the addressing scheme already introduced to implement the crum structure and enfilade.

MULTIDIMENSIONAL BEDS

10 The roving operation is not limited to one-dimensional data, as the superficially one-dimensional stream structure would suggest. Rather, it may be extended without difficulty to two-dimensional, three-dimensional and multidimensional data forms. That is, the stream, bed and babble systems expand readily to the representation of two-dimensional and higher-dimensional structures, providing multidimensional views of these structures in core, incrementally movable through the multidimensional structures with constant
15 core addressing.

New dividers, which we may call n-dividers, specify the dam positions in the separate dimensions: the 1-divider in dimension 1, ... the n-divider in dimension n. However, the overall view can no longer be incrementally babbled as before. Only one dimension may be babbled incrementally
20 and contiguously. To babble in another dimension, combinations of beds must be wholly rebabbled, at a greater expenditure in time. To maintain an even and symmetrical window on the data-- exceptions, while interesting, will not be considered here-- it is necessary to babble by one complete increment in a given dimension; however, each dimension is differently defined.

25 In the two-dimensional case, we may define a file as a series of parallel streams, each representing a row (or column) of a much larger checker-board array. We may also segment the data correspondingly, so (letting "/"

represent dividers and "." divide between rows or columns) we may have:

A4, A5 / A1, A2, A3 . B4, B5 / B1, B2, B3 . C4, C5 / C1, C2, C3 .
D4, D5 / D1, D2, D3

Thus we may babble in two separate dimensions, which we may here call
5 "consecutive" and "broken," depending on whether babble is incremental or
scattered. Consecutive babble:

E4, E5 / E1, E2, E3 replaces A4 ... A3.

Broken babble (from four streams):

A4, A5, A6, A7 / A3 . B4, B5, B6, B7 / B3 . C4, C5, C6, C7 / C3.
10 D4, D5, D6, D7 / D3

We may also think of this as having two dividers, a "long divider" between
sections actually consecutive in streams, and a "short divider" within each
set of elements having to be broken-babbled.

A A | A A A | B B | B B B . C C | C C C . D D | D D D

15 This is simply seen as equivalent to

A A		A A A
B B		B B B
C C		C C C
D D		D D D

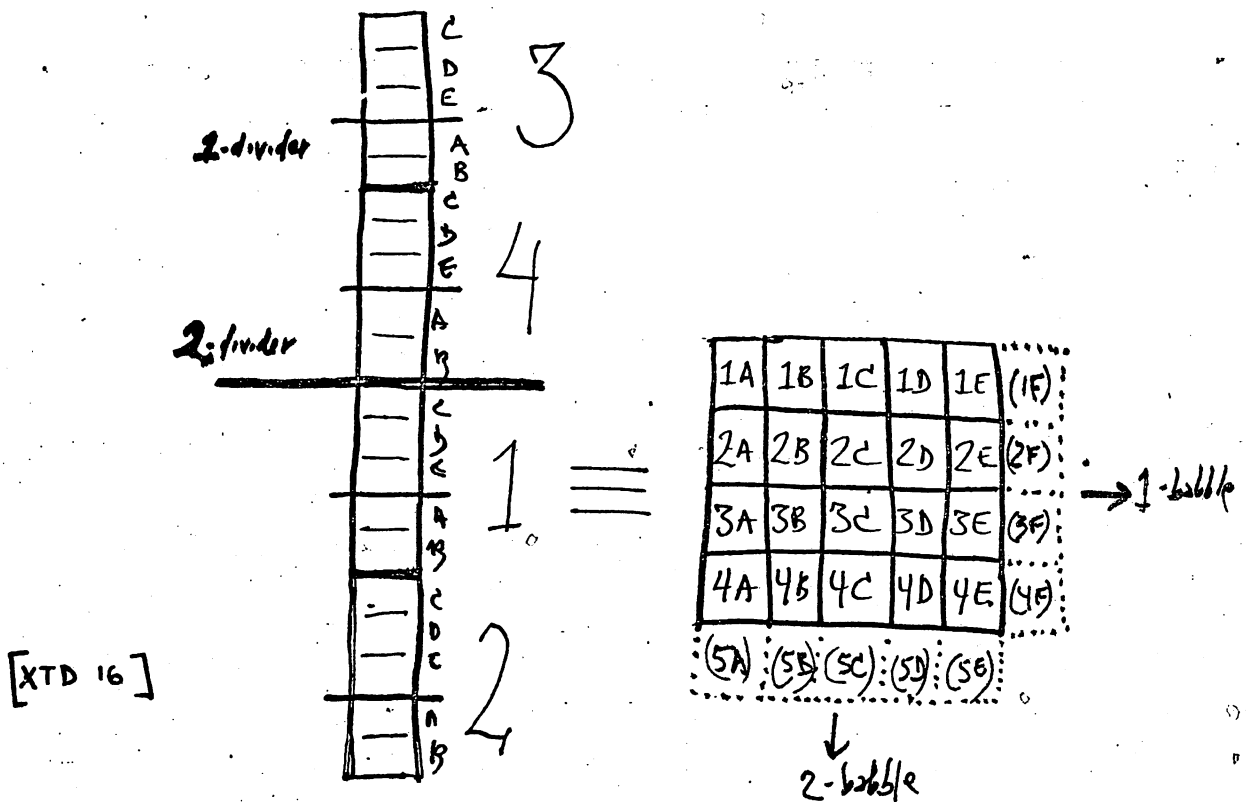
20 Or, considered as a moving window on a large checkerboard, the above doubly-
numbered example inverts to:

A3		A4 A5 A6
B3		B4 B5 B6
C3		C4 C5 C6
D3		D4 D5 D6

25 We may also visualize it as follows:

WFC

64



The three-dimensional case extends itself obviously, merely adding another divider:

// A13, A14, A15 / A16 . B13, B14, B15 / B16 . C13, C14, C15 / C16

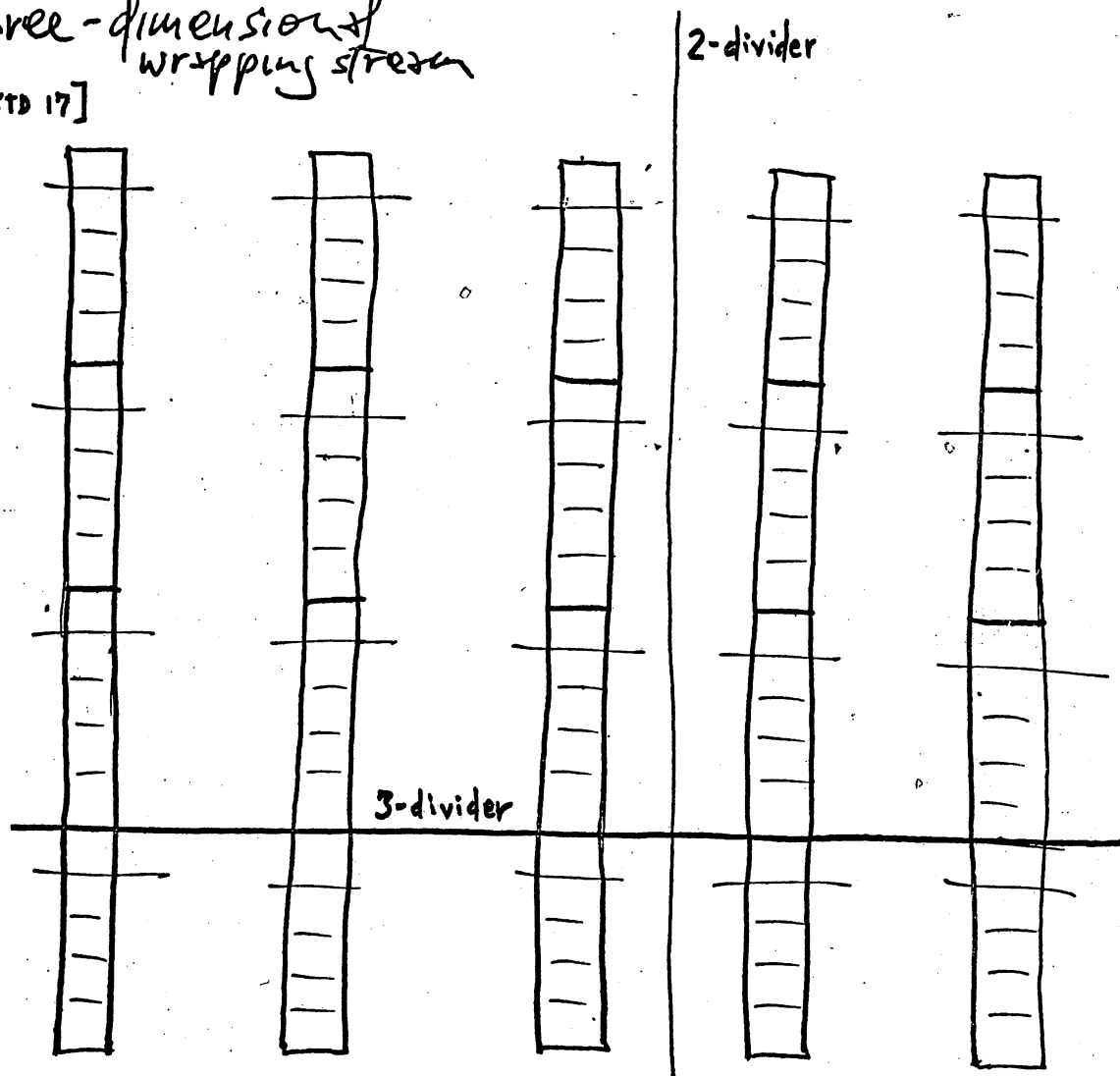
// A23, A24, A25 / A26 . B23, B24, B25 / B26 . C23, C24, C25 / C26

// A33, A34, A35 / A36 . B33, B34, B35 / B36 . C33, C34, C35 / C36

In this instance roving may be three-dimensional, regarding the aslash, double-slash and vinculum in this arrangement each as a divider which may babble.

This has many obvious uses, such as the portrayal of three-dimensional architecture and the representation of clustered statistical data on three variables.

three-dimensional
wrapping stream
[XTD 17]



The Parallel Textface is considered exemplary to the operation of this invention because of its generality and complexity. However, certain other examples will be given of useful employments of the system.

"The Bicyclist"

It was earlier stated that this invention provides a generalized structure for the transfer of pictorial and textual information. An example may aid in comprehending this.

5 Suppose that the user of the system is reading an article (or a story) in which bicycles are mentioned. As he throttles forward, and the text slides up on the screen, an illustration of a bicyclist can appear on the screen, with his legs, and the spokes of the wheels, going around rhythmically. As he reaches the top of the screen he stops pedalling, dismounts momentarily, stands still, and then begins to tap-dance.

10 Several mechanisms are employed to get this effect, which is here selected to show several important aspects of the system.

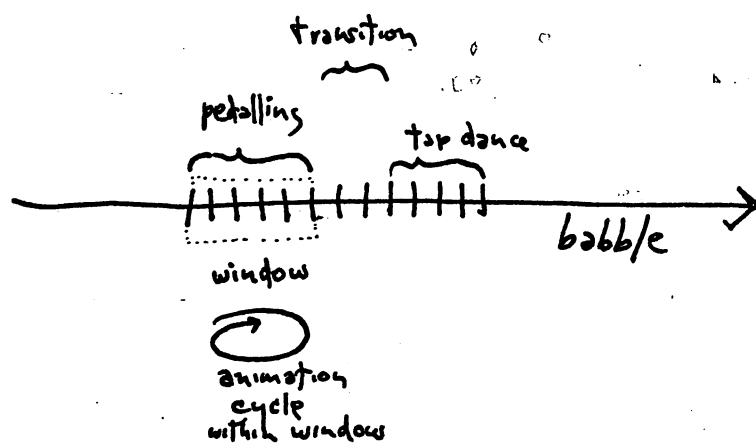
15 The spokes of the wheel are handled by separate frames showing a succession of radial lines, in one quadrant of the wheel, in animation sequence.



Sign-changing flips are employed (bits 13-15 of the DINGO "DISPLAY" instruction) to fill out the circle by replication of each.

The pedalling sequence is a cycle of frames showing leg positions.

The top-of-the-screen sequence is another series of frames, babbled in to replace the previous sequence.



Accordingly, the stream structure is

DINGO

Screenmaster

Part Map

Picmaster

Picture

TM

T

Expandable Map

It is generally desired to present a map on the screen which may be magnified to any degree, and slid in any compass direction, whether at greatest or least magnification or between.

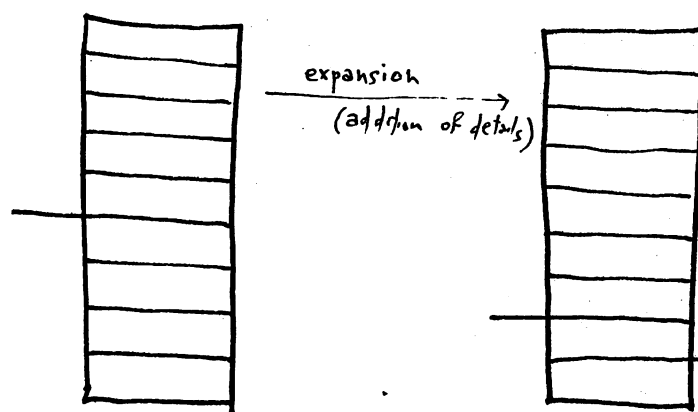
5 This is accomplished in the present invention by a pair of matched 2-dimensional beds holding matched map-line structures at two successive levels of magnification.

10 The user may command compass movement with the light-pen, e.g., on a compass-rose, and the "zoom" in or out by means of either-directional motion on a control line.

15 The DINGO program examines the differential position of the controlling light pen and shrinks or expands the picture accordingly, scissoring the vectors (through conventional techniques) as required. Only those vectors or partial vectors are shown which fit into the given map window at the current level of magnification.

Bed for upper level

Bed for deeper level



Paired 2-dimensional beds
(when right one assumes full control,
left one is replaced by next
deeper stream and expansion
now continues into left bed.)

Map 1

70

The stream structure is:

Shallow (globe level)

Deeper (national level)

Still deeper (state level)

Deeper yet (county level)

Deepest (street level)

Uniquely in this system, 2-dimensional babble permits the user to rove in any compass direction indefinitely at any level of magnification, concurrently changing magnification level at will.

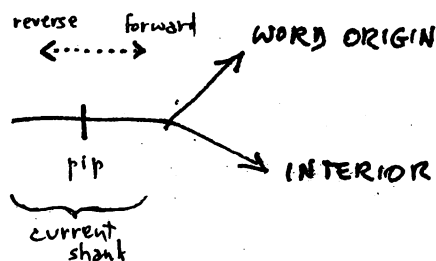
Walking Nettm Picture Control System

Another application of the present invention is also in the area of pictorial display, but offers a great variety of potential user choices in a simple circumstance. I call this the "walking net" system because control is effected through a changing network of choices which step, or "walk," around the screen.

The problem of intricate computer graphics may be phrased as follows: given that a digital system can hold a wide variety of graphical materials ready to present, how may the user most simply and conveniently choose them? Indeed, how may the user keep track of what is happening, where he is and where he has been?

The external mechanism I have selected for this facility paradoxically combines great versatility for sophisticated presentations with great simplicity before the naive user. The idea is this: the user may command a continuing succession of changing presentations, making only one simple choice at a time, yet receiving intricate and rich animations with extremely clear continuity on the screen.

The exterior mechanism is this: along with an arbitrary graphic presentation on the screen, the user is continuously presented with the image of a forking set of arrows, e.g.:



WNet 1

72

The pip is a conventional light-pen cursor. The "current shank" is a line whose implicit gradations control developments in the picture; and the choice of arrows at the end of the current shank determine a discrete choice between alternatives that are to transpire.

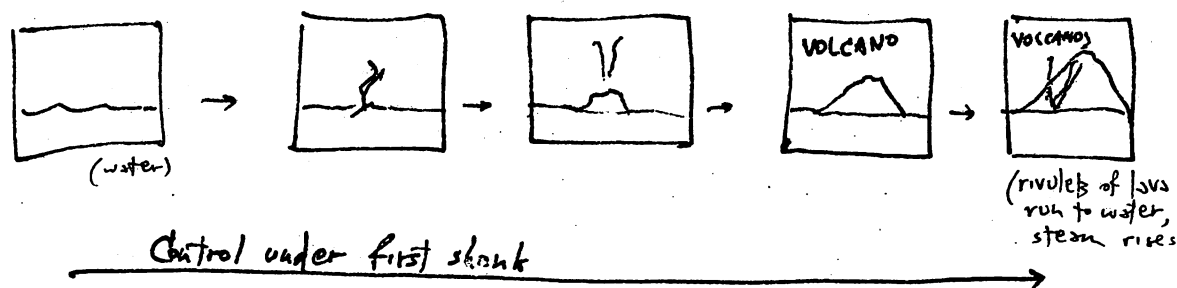
5 The user, seizing the pip with the lightpen, moves it (through the usual lightpen techniques) sideways along the current shank. Moving it in the "forward" direction causes progressive developments in the picture, moving it "backward" causes a reversal of animations and other previous developments.

10 When the pip reaches the choice point in the forward direction, the user may drag it (through the usual lightpen techniques) along either of the beckoning alternatives. This then causes further developments in the presentation consonant with the line selected.

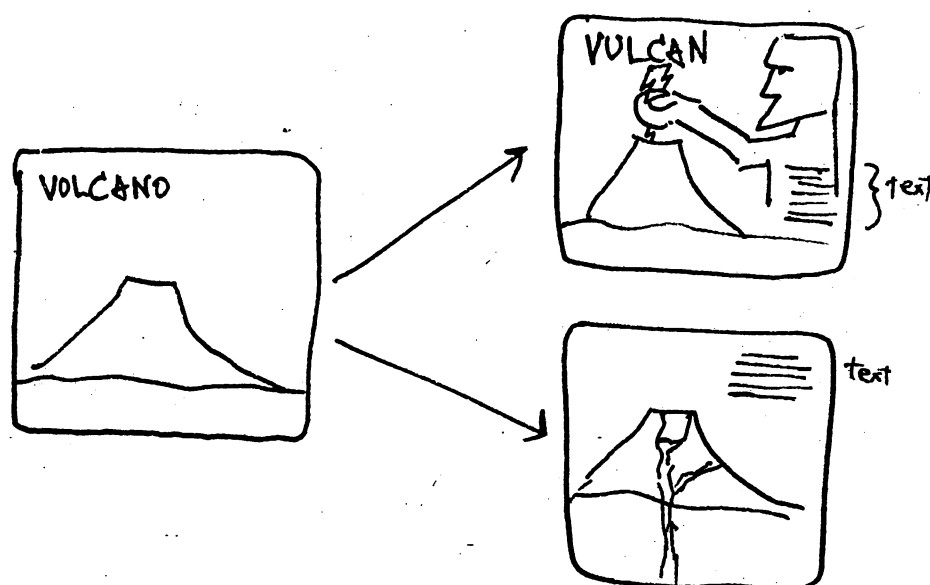
"Developments" of the picture here include expansion, contraction, sliding movements and frame-by-frame animation.

15 (These materials will have been, of course, explicitly input by authors and artists.)

20 In a sample employment, consider a presentation on the subject of Volcanoes. Let the first shank of the control net control the "rise of a volcano from the sea"-- an undulating ocean surface pierced first by a wisp of smoke, then a growing peak, with rivulets of lava seen to run down its sides and darken as they contribute to its growth.



At the end of the first shank, the user may branch to two arrows, labelled respectively WORD ORIGIN and INTERIOR. Either option continues the presentation without a break, retaining much of the picture on the screen. Selection of WORD ORIGIN causes the word VOLCANO to change to VULCAN, and a picture of the god Vulcan is seen to seize a lightning bolt rising from the crater; text appears to explain this. Alternatively, if the user chooses INTERIOR, the tubes and ducts within the volcano appear, and explanatory text also.



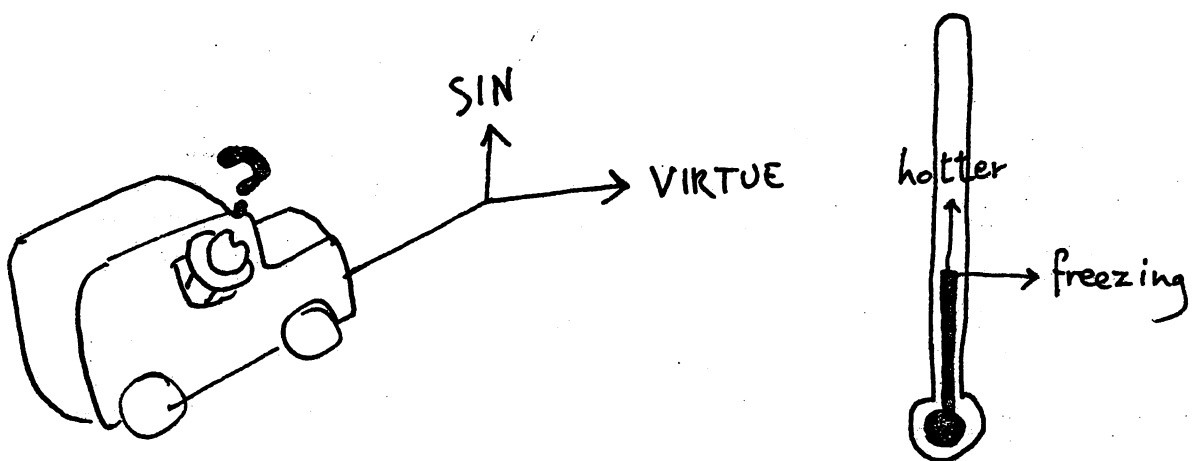
(The path unchosen fades from the screen, as does the previous shank.)

Either of these alternatives may continue with its own developments and animations under control of its own shank.

Several features of this control application are of special interest. One is that the presentation may be continuous in all directions, aiding in continuous user orientation. Another is that presentations are reversible in various ways, an aid both in user orientation and self-study.

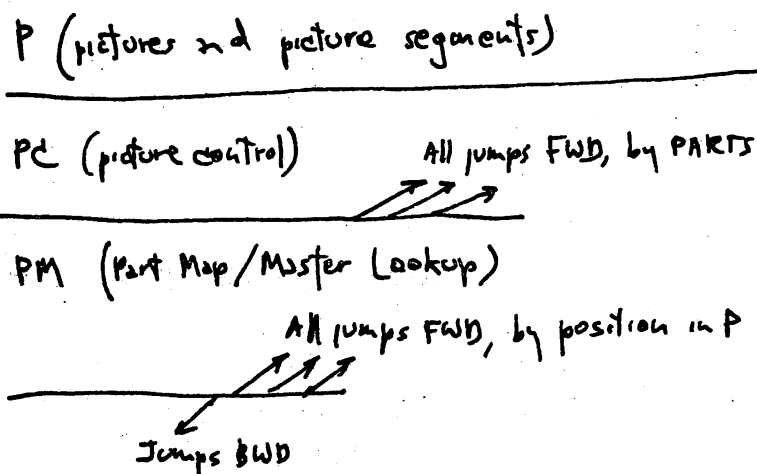
5 (Not only is a demonstration reversible within a given shank, but the user may back the pip through an intersection into the antecedent shank-- which reappears at the juncture as the lightpen backs up-- and the user may continue to reverse the presentation through that preceding shank, or to re-enter the intersection and make another choice, "the path not taken.") These
10 features allow the user clearly to repeat demonstrations as often as he likes and to explore numerous alternatives.

The displayed control net is thus to be understood as a large network of choices, mostly unseen, whose currently visible portion "walks" around the screen as use progresses. Within this system, then, numerous variants are
15 possible. For instance, the currently visible portion of the net may itself be whimsically incorporated in a picture, viz.:

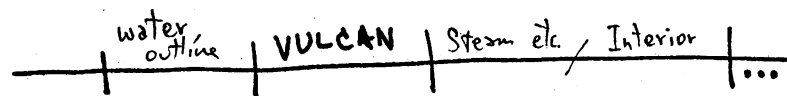


To accomplish this Walking Net system, this invention's same internal techniques are employed in a different configuration. Animation and picture-development techniques are all implicit in the TC/PM codes and the structure of the DINGO language. It remains for us only to consider here how the search and buffering techniques of the present invention contribute to the desired result, from which construction techniques for particular presentations will be obvious to any skilled practitioner of the arts of digital computer animation.

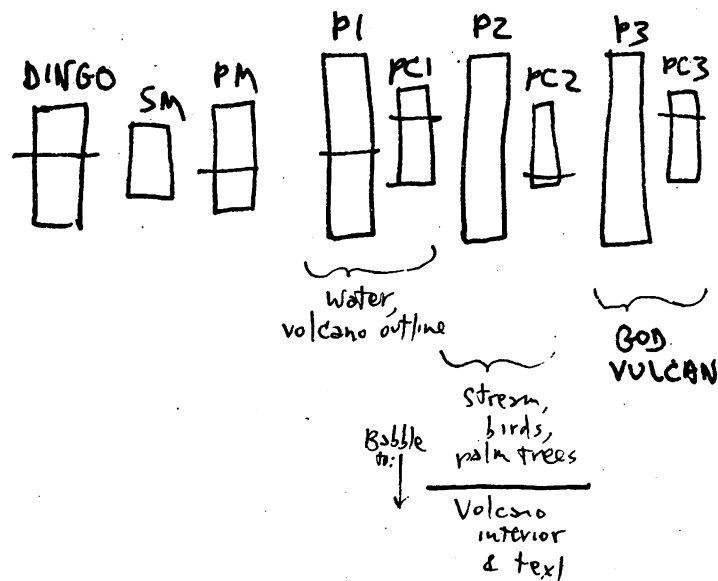
A simplified stream-structure is employed, having only three streams:



However, picture-stream P is segmented in such a way as to make babblable picture-sections consecutive and contiguous:



Thus the bed layout has several "P" and "PC" beds to permit simultaneous buffering and presentation of different sections of the walking net:



Screenmaster SM is here used by DINGO principally for the storage of current screen-values, such as locations on the screen which must be commonly addressed by two parts of a picture, and locations of node-coordinates for the displayed segment of the control net.

OPERATION

The succession of devices and structures described, set into operation, effects the several purposes of the system through a combination of techniques.

5 The system has several important modes of operation, to be described in turn. Foremost are those associated with the operation of the Parallel Textface, particularly in its screen motion with displayed linkages; but this rests on the performance of all the other subfunctions of the system. We refer to FIG. 6.

10 Search Microprocessor SE P is seen to consist of Read-Only Memory ROM1 connected to Rapid-Access Memory RAM and Crum Level Indicator Table CLIT. Rapid-Access Memory RAM is also connected to Crum-Level Indicator Table CLIT, and to Crum Table CT, Data Buffer DB, Select register SL, and Memory Address Register MAR.

15 Select Register SL is connected to Floppy Disk FD and Cassette CST. Data Buffer DB is connected to Floppy Disk FD and cassette CST; information is babbled in from them, in addition to contents of CT and CLIT which hold crums and their crum-pointers for all streams. (In a significant variation, CT and/or CLIT may be cycled for use among different streams, by
20 swapping the contents of CT and CLIT to disk by the usual swapping means.

 Data Buffer DB is also connected to Digital Core Memory DCM, to which it furnishes babbled input.

 Rapid-Access Memory RAM stores dividers and temporary operands required to work with crums in Crum Table CT and Crum Level Identifier Table
25 CLIT.

Digital Computer Memory ICM contains, as a significant part of its control circuitry, its Memory Address Register MAR specifying core locations to which information is being babbled from Floppy Disk FD, Cassette CST and Rapid-Access Memory RAM. Addressable locations of Digital Core Memory DCM, here
5 illustrated as subdivided for execution of the Parallel Textface, is partitioned into beds enacting the behavior of the first text: Bed T1 holding Text 1, Bed TC1 holding Text Control 1, Bed PM1 holding Part Map 1, Bed CPM 1/2 holding Counterpart Map 1/2; beds enacting the behavior of the second text: bed T2 holding Text 2, Bed TC2 holding Text Control 2, Bed PM2 holding Part Map 2,
10 Bed CPM 2/1 holding Counterpart Map 2/1; beds enacting the behavior of links: Bed LL holding left broken links, Bed LR holding right broken links, and Bed LC holding completed links; buffer CMRB the Crum/Meander Repacking Buffer; Screenmaster SM; Furniture List F; and DINGO program for Parallel Textface PTF.

User responses and control actions undertaken with Lightpen LP
15 and keyboard KB return to Search Microprocessor SE P via keyboard buffer KBB and lightpen buffer LPB, where it initiates action by microprogram lookup in Read-Only Memory ROM1 (if, e.g., JOT commands) or passes transparently through to RAM (if, e.g., text being inserted).

20 Counts required for character or figure displays are transferred from DINGO programs or count beds (such as TM), then controlling the transfer of picture elements through vector pbuffer VB to CRT control registers OX and OY, thus to cathode-ray tube CRT, or character buffer CB to character generator CG and thus to cathode-ray tube CRT. During the time-slice here shown, text in bed T1 is being displayed serially under count control of textmaster counts in,
25 bed TML. Furniture F is also being shown, and material is being babbled into

79

~~79~~

bed T1.

DINGO program for Parallel Textface PTF goes to interpretation lookup table in Read-Only Memory ROM2, by lookup directing subinstructional sequence in Read-Only Memory ROM3. Temporary Operands and "pushed" commands
5 are placed on stack STK, from which they are popped as required.

Vectors from furniture bed F are transferred to Vector Buffer VB, whence it is transferred to CRT control registers OX and OY, effecting the presentation of line-drawings.

Search and Retrieval

Theory. The principle is that search on a desired stream position proceeds downward through the enfilade, counting WIDs and rejecting those which sum with their left cousins and brothers to too little or too much to yield the desired stream position.

Crum-blocks, when called, arrive in buffer CMPB; crums saved from a crum-block are then moved by microprocessor SE_MP to crum table CT, being sandwiched in their place according to Ancestral Order. Left-out left brothers are replaced by SE_MP with a left-out pseudo-crum.

Crum-blocks are repacked as needed in CMPB.

For stream movement, especially as required by BABBLE, final meanders are called sideways, proceeding in a leftward or rightward direction, jumping to cousins when brothers run out.

The techniques of roving and babble are executed according to the procedures already disclosed. Referring particularly to FIG. 14, we see that in forward babble we (A) replace element F with the next stream element, and test whether that element F was foot F^t , if not return to (A); if so reset divider to location of head H and return to (A).

Babble in the backward direction is effected by setting a temporary divider to the anticipated location of the divider on completion of the babble, also holding said temporary divider in temporary location TEMP (expository label for current description only). We then proceed with normal forward babble, moving the temporary divider. When the temporary divider equals the real divider, back-babble is complete and the true divider value is obtained from TEMP.

The principal inner operations of Search Microprocessor SE_μP will now be detailed.

As previously stated, microprocessor SE_μP enacts the operations of search, retrieval and editing according to the principles already set forth. The following procedure-set, encoded into microinstructions in read-only memory of Search Microprocessor SE_μP, manages the editing and retrieval functions for the enfilade system that has been expounded. (The detailed operation of these elements is given in the Algol language, the universal language of procedure description. The Algol dialect employed is that of the Data General Corporation, which has the advantage of allowing the declaration of bit-field, string and label variables and arrays, as well as the usual integer and real variables and arrays.)

It will be noted that a crum is referred to by its FAMILYTREE, an expression of its exact position in the enfilade as a vector of ordinal positions from the top down to its level. The family tree is the general referent employed for a crum because the crum's specific CT position is assumed to be constantly changing. FAMILYTREE is a vector of integers expressing its ordinal position of descent from the zero-crum down. Thus the family tree of the seventh bottom crum in FIG. 8 is 0, 4, 4, 3, 0, 0.

A crum's effective identifier at a given instant is CLT SUBSCRIPT, which specifies its position in the CT and its marker's position in the CLIT. (CT and KT are the crum table addressed by different rules of subdivision.)

The CLIT bits keep track of running operations. DROPBIT indicates that a crum and all its children are to be destroyed. CHANGEBIT indicates that this crum has been changed. Operation CLEANUP (not shown) repacks the crums as ordained by DROPBIT and CHANGEBIT.

The procedures here given call each other and are called by other programs, notably BABBLE and the edit systems, notably including JOT.

1. Housekeeping Operations: FAMILYTREE, FFATHERCRUM, CRUMPOSITION, FINDTWIG, ZAPOFFSPRING, KILLCRUMS.

COMMENT: PROCEDURE TO BUILD A FAMILY TREE
CALLING SEQUENCE
FAMILYTREE (ARRAY, SUBSCRIPT OF CRUM, WORD, ELABEL)
THE ARRAY SHOULD BE LARGE ENOUGH TO CONTAIN
THE MAXIMUM NUMBER OF LEVELS
WORD IS RETURNED WITH THE LEVEL # OF THE CRUM
WHICH IS THE HIGHEST SUBSCRIPT OF THE ARRAY
THE ARRAY POSITIONS ARE FILLED IN WITH POSITION NUMBER
AS THE FATHER SEES THEM;

```
BEG. PROCEDURE FAMILYTREE (ARR, SUBSC, RLEVEL, ELABEL)
  VALUE SUBSC;
  ARRAY ARR;
  INTEGER ARR, SUBSC, RLEVEL, I, J;
  LABEL ELABEL;

  RLEVEL := CLT(SUBSC) AND LEVELMASK;
COMMENT: GET LEVEL OF CRUM;

  I := SUBSC;

  FOR J := RLEVEL TO 1 STEP -1 DO BEGIN
    ARR[I] := CRUMPOSITION (I, ELABEL);
COMMENT: GET POSITION OF CRUM;

    I := FFATHERCRUM(I, ELABEL);
COMMENT: GET SUBSCRIPT OF FATHER CRUM;
  END;
  ARR[0] := 1; IF LBOUND (ARR,1) < 0 THEN ARR[-1] := RLEVEL;

  END;
```

SR5

48

83

COMMENT: PROCEDURE TO FIND TO THE FATHER CRUM OF ANY GIVEN CRUM
 CALLING SEQUENCE FFATHERCRUM(I,KLABEL))
 WHERE "I" IS THE SUBSCRIPT OF THE CLT WORD
 FATHERCRUM IS THE SUBSCRIPT ANSWER-- LABEL IS FOR
 ERROR RETURN;

```
BEGIN INTEGER PROCEDURE FFATHERCRUM(I,KLABEL);
  INTEGER I, LEVEL, LEVEL2;
  LABEL KLABEL;
  VALUE(F);
  LEVEL := CLT(I) AND LEVELMASK;
  FOR J := I-1 TO 0 STEP -1 DO BEGIN
    LEVEL2 := CLT(I) AND LEVELMASK;
    IF LEVEL2 < LEVEL THEN GO TO J1; END;
  J1:FFATHERCRUM := J;
  IF LEVEL2 <> LEVEL -1 THEN GO TO KLABEL; END;
```

COMMENT: PROCEDURE TO FIND POSITION # OF A CRUM
 CALLING SEQUENCE-- CRUMPOSITION(SUBSCRIPT[,LABEL])
 LABEL IS FOR ILLEGITIMATE FATHER;

```
BEGIN INTEGER PROCEDURE CRUMPOSITION(I,KLABEL);
  VALUE I; LABEL KLABEL;
  INTEGER I, J, J2, J3, J4;
  J := FFATHERCRUM(I,KLABEL);
  J2 := 0;
  J3 := CLT(I) AND LEVELMASK;
  FOR J4 := J+1 TO I STEP 1 DO
    IF CLT(J4) AND LEVELMASK = J3 THEN
      IF CLT(J4) AND LEFTOUTBIT = 0 THEN
        J2 := J2 + 1 ELSE J2 := J2 + (CLT(J4) AND HW1);
  CRUMPOSITION := J2;
  DONE: END;
```

COMMENT: PROCEDURE TO SUPPLY CLT SUBSCRIPT GIVEN A FAMILY TREE
AND SUBSCRIPT (LEVEL) WANTED
CALLING SEQUENCE.
FINDTWIG (ARRAY, SUBSCRIPT, ELABEL);

BEGIN INTEGER PROCEDURE FINDTWIG (ARR, SUBSC, ELABEL)
ARRAY ARR; VALUE ARR, SUBSC; LABEL ELABEL;
INTEGER ARR, SUBSC, K, JS, LEVEL, CNT;

K := -1;
FOR I := 0 TO SUBSC STEP 1 DO BEGIN
JS := K + 1;
J := ARR[I];
CNT := 0;
FOR K := JS TO CLTMAX STEP 1 DO BEGIN
LEVEL := CLT[K] AND LEVELMASK;
IF LEVEL := I THEN IF CLT[K] AND LEFTOUTBIT = 0 THEN
CNT := CNT + 1 ELSE CNT := CNT + (CLT[2, K] AND HW1) ELSE
GO TO 1; IF CNT => JS THEN GO TO 2;
1: IF LEVEL < I THEN GO ELABEL;
2: END;
FINDTWIG := K; END;

COMMENT PROCEDURE TO ELIMINATE THE CHILDREN OF A CRUM
CALLING SEQUENCE ZAPOFFSPRING(SUBSCRIPT);
BEGIN

```

PROCEDURE ZAPOFFSPRING(I);
VALUE I, J2, J4, J3, J5, J;
INTEGER 1;
J4 := 0;
J2 := CLT(I) AND LEVELMASK + 1;
FOR J: I+1 TO CLTMAX STEP 1 DO BEGIN;
IF CLT(J) AND LEVELMASK <= J2 THEN GO 1;
1: IF J4: 0 GO TO DONE;
FOR J3 := I + 1 TO CLTMAX STEP 1 DO BEGIN
FOR J5 := 1 TO 4 STEP 1 DO CT(J5, J3) := CT(J5, J3+J4);
CLT(J3) := CLT(J3+J4); END;
CLTMAX := CLTMAX - J4; DONE; END;

```

COMMENT: PROCEDURE TO ELIMINATE A CRUM AND ITS CHILDREN
CALLING SEQUENCE-- KILLCRUMS(I)
WHERE "I" IS THE SUBSCRIPT OF THE CLT WORD;
BEGIN PROCEDURE KILLCRUMS(I); INTEGER I, J, J1, LEVEL,
LEVEL2;

VALUE I;

COMMENT: IF THE LEFT ADJACENT CRUM IS A "LEFT OUT" CRUM, THEN
THE WID & THE LEFT OUT COUNT ARE ADJUSTED ACCORDINGLY;

```

LEVEL := CLT(I) AND LEVELMASK;
CLT(I) := CLT(I) OR DROPBIT;
FOR J := I+1 TO CLTMAX STEP 1 DO BEGIN
LEVEL2 := CLT(J) AND LEVEL MASK;
IF LEVEL2 <= LEVEL GO TO LFIN;
CLT(J) := CLT(J) OR DROPBIT; END;
LFIN: IF CLT(I-1) AND DELTABIT = 0 GO TO LFIN2;
KT(2, I-1) := KT(2, I-1) + KT(2, I);
CT(2, I-1) := CT(2, I-1) + 1;
J1 := J-1;
LFIN2: FOR K := 1 TO CLTMAX STEP 1 DO BEGIN
CLT(K) := CLT(K+J1);
FOR K1 := 1 TO 4 STEP 1 DO CT(K1, K) := CT(K1, K+J1); END;
CLTMAX := CLTMAX - J1;
J1 := FFATHERCRUM(I);
CT(2, J1) := CT(2, J1) - 1; END;

```

2. Crum-cutting operations: CRUMCUT, TEXTSPLIT, CRUMSPLIT.

COMMENT: PROCEDURE TO CUT A CRUM WHICH POINTS TO ANOTHER CRUM
CALLING SEQUENCE
CRUMCUT(I,J)

WHERE I = SUBSCRIPT #

J = # CRUMS GOING TO FIRST CRUM;

```
BEGIN PROCEDURE CRUMCUT(SUB, NU);
ARRAY A(40)
VALUE SUB,NU; INTEGER(2) WID,WID2,LOC;
INTEGER SUB,NU,F,J2,J,F2,K2,K3,K4,K5,K,K6,A;
COMMENT: F=LEVEL OF CHILDREN;F:=(CLT[SUB] AND LEVELMASK)+1;
J2:=WID:=J3:=0;
FOR J:=SUB +1 TO CLTMAX STEP 1 DO
IF CLT[J] AND LEVELMASK =F THEN BEGIN
COMMENT: WID=SUM OF CHILDREN'S WIDS;
WID:=WID + KT[2,J];
IF CLT[J] AND LEFTOUTBIT=0 THEN J2:=J2+1 ELSE J2:=J2+CT[2,J]
AND HW1;
IF J2:=>NU THEN GO TO 1;
END;
1: WID:2 := KT[2,SUB] -WID
LOC:=NEXTLOC;
NEXTLOC:=NEXTLOC+1;
FOR K:=J+1 TO CLTMAX STEP 1 DO
IF CLT[K] AND LEVELMASK = F THEN GO TO 2;
2: F2:=F+1;
FOR K2:=K+1 TO CLTMAX STEP 1 DO
IF CLT[K2] AND LEVELMASK <=F THEN GO TO 3;
3: FOR K3 :=K TO K2 STEP 1 DO
IF CLT[K3] AND LEVELMASK =F2 THEN BEGIN
IF CLT[K3] AND LEFTOUTBIT <>0 THEN BEGIN
LOCPOSS(K,A);
I:=(CT[2,K3] AND HW1) -1;
COMMENT: OPEN A HOLE FOR LEFT OUT CRUMS;
FOR K4:=CLTMAX TO K3 STEP -1 DO BEGIN
K6:=K4 +I;
CLT[K6]:=CLT[K4];
FOR K5:=1 TO 4 STEP 1 DO CT[K5,K6]:=CT[K5,K4]; END;
CLTMAX:=CLTMAX +I;
LOCPOSS(A);
FILLREAD(K,NU+J3,A); END;
J3:=J3+I;
FOR K5:=K3 TO K3+I STEP 1 DO
CLT[K5]:=F2 OR CHANGEKIT;
IOCOM(A); END; ELSE J3:=J3+1; END;
CRUMSPLIT(-K2,1);
KT[2,K2]:=WID2
CT[1,K2]:=LOC
CT[2,K2]:=J3+(CT[2,J] AND BANKMASK) + SHIFT(SHIFT(LOC,-16),8)
CLT[K2]:=F OR CHANGEKIT
END;
END;
```

COMMENT: PROCEDURE TO CUT A CRUM WHICH POINTS TO TEXT
 CALLING SEQUENCE
 TEXTSPLIT(I,DWID)
 WHERE I = SUBSCRIPT OF CRUM
 DWID - # CHARS TO GO TO FIRST CRUM;

```
BEGIN PROCEDURE TEXTSPLIT(I, DWID, MODL, MODR)
VALUE I, DWID, MODL, MODR; INTEGER(2) DWID, DWID2, LOC
INTEGER I, J, V, V2;
CRUMSPLIT(I,1);
J := I+1;
KT(2,J) := KT(2,I) - DWID;
KT(2,I) := DWID;
DWID2 := DWID / 128;
LOC := SHIFT(CT(2,I),-8) AND 15;
V2 := LOC;
LOC := SHIFT(LOC,16) + CT(1,I) - DWID2;
CT(1,J) := LOC;
V := CT(2,I);
CT(2,J) := V AND K70000 + DWID - DWID2 * 128 + SHIFT(V2,8);
CLT(5) := CLT(J) OR MODR OR CHANGEBIT;
CLT(1) := CLT(I) OR MODL OR CHANGEBIT;
END;
```

COMMENT: PROCEDURE TO SPLIT A CRUM
 CALLING SEQUENCE
 CRUMSPLIT(I,12);

```
BEGIN PROCEDURE CRUMSPLIT(I,12);
VALUE I,12, INTEGER I,12,J,J2;
CLTMAX := CLTMAX + 12;
FOR J := CLTMAX TO ABS(I) + 12 -1 STEP -1 DO BEGIN
FOR J2 := 1 TO 12 STEP 1 DO CT(J2,J) := CT(J2,J-12);
CLT(J) := CLT(J-12); END;
IF I < 0 THEN GO TO DONE
J2 := FFATHERCRUM(I);
CT(2,J2) := CT(2,J2) + 12;
DONE: END;
```

8028

88

3. Edit executions: EXECUTEDELETE (same as DELEX in JOT), EXECUTESWITCHEROO
(same as SWEX in JOT).

COMMENT PROCEDURE TO EXECUTE A DELETE
CALLING SEQUENCE

EXECUTEDELETE;

BEGIN PROCEDURE EXECUTEDELETE;

ARRAY ARR1[-1:31]; ARR2[-1:31];

INTEGER I, I2, J, ARR1, ARR2, J1, LVL1, LVL2;

FOR I := 1 TO CLTMAX DO IF CLT [I] AND DELCUTMASK > 0
THEN GO TO 1;

1: FOR I := 1 TO CLTMAX DO IF CLT[I] AND DELCUTMASK > 0
THEN GO TO 1;

2: FAMILYTREE (ARR1, I, LVL1);

FAMILYTREE (ARR2, I2, LVL2);

FOR J := 0 TO IF LVL1 > LVL2 THEN LVL2 ELSE LVL1 DO
IF ARR1[J] <> ARR2[J] THEN GO TO 3;

3: FOR J1 := LVL1-1 TO J STEP -1 DO

CRUMCUT(FINDTWIG(ARR1,J1), ARR1[J1+1], 0, DROPBIT);

FOR J1 := LVL2-1 TO J STEP -1 DO

CRUMCUT (FINDTWIG(ARR2,J1) ARR2[J1+1], DROPBIT,0);

END;

SR9

89

```

COMMENT PROCEDURE TO EXECUTE A SWITCHEROO
CALLING SEQUENCE
EXECUTESWITCHEROO;
BEGIN PROCEDURE EXECUTESWITCHEROO;
  INTEGER ARRAY ARR1C-1:31J, ARR2C-1:31J, ARR3C-1,31J;
  INTEGER I, I2, I2, JMIN, LVL1, LVL2, LVL3, J, J1, J2, J3, J4,
  J5, J6;

  FOR I := 1 TO CLTMAX DO IF CLTI1 AND XCUTMASK <> 0 THEN GO TO 1;
  1: FOR I2 := 1 + 1 TO CLTMAX DO IF CLTI2 AND XCUTMASK <> 0 THEN GO TO 2;
  2: FOR I3 := I2 + 1 TO CLTMAX DO IF CLTI3 AND XCUTMASK <> 0 THEN GO TO 3;
  3: FAMILYTREE(ARR1, I, LVL1);
  FAMILYTREE(ARR2, I2, LVL2);
  FAMILYTREE(ARR3, I3, LVL3);
  JMIN := IF LVL1 > LVL2 THEN LVL2 ELSE LVL1;
  JMIN := IF LVL3 > JMIN THEN JMIN ELSE LVL3;
  FOR J := 0 TO JMIN DO BEGIN;
    IF ARR1CJ := ARR2CJ = ARR3CJ THEN GO TO 4;
    GO TO 5; 4: END;
  5: FOR J1 := LVL1 - 1 TO J STEP -1 DO
    CRUMCUT(FINDTWIG(ARR1, J1), ARR1CJ1 + 1J, 0, 0);
    FOR J1 := LVL2 - 1 TO J STEP -1 DO
    CRUMCUT(FINDTWIG(ARR3, J1), ARR3CJ1 + 1J, 0, 0);
    J2 := J - 1;
    CRUMCUT(FINDTWIG(ARR2, J2), ARR2CJ2 + ARR2CJ1, 0, 0);
    CRUMCUT(FINDTWIG(ARR1, J2), ARR1CJ2, 0, 0);
    ARR2CJ2 := ARR2CJ2 + 1;
    ARR3CJ2 := ARR2CJ2 + 1;
    J3 := FINDTWIG(ARR2, J2);
    J4 := FINDTWIG(ARR3, J2);
    J5 := CLTIJ3;
    CLTIJ3 := CLTIJ4; CLTIJ4 := J5;
    FOR J6 := 1 TO 4 STEP 1 DO BEGIN J5 := CLTIJ6, J3;
      CLTIJ6, J3 := CLTIJ6, J4; CLTIJ6, J4 := J5; END; END;

```